МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

Федеральное государственное автономное

образовательное учреждение

высшего образования

«Национальный исследовательский университет ИТМО»

(ФГАОУ ВО «ИТМО»)

Факультет: «Факультет программной инженерии и компьютерной техники»

Дисциплина: «Программирование на Java».

Лабораторная работа № 3

Вариант № 32499

**Группа** P3121

**Выполнила:**

Шукалович Екатерина Андреевна

**Проверила:**

Заболотняя Ольга Михайловна

Санкт-Петербург 2025

**Цель:** В соответствии с выданным вариантом на основе предложенного текстового отрывка из литературного произведения создать объектную модель реального или воображаемого мира, описываемого данным текстом. Должны быть выделены основные персонажи и предметы со свойственным им состоянием и поведением. На основе модели написать программу на языке Java.

**Этапы выполнения работы:**

1. Получить вариант
2. Нарисовать UML-диаграмму, представляющую классы и интерфейсы объектной модели и их взаимосвязи;
3. Придумать сценарий, содержащий действия персонажей, аналогичные приведенным в исходном тексте;
4. Согласовать диаграмму классов и сценарий с преподавателем.
5. Написать программу на языке Java, реализующую разработанные объектную модель и сценарий взаимодействия и изменения состояния объектов. При запуске программа должна проигрывать сценарий и выводить в стандартный вывод текст, отражающий изменение состояния объектов, приблизительно напоминающий исходный текст полученного отрывка.
6. Для одного из методов класса персонажа (или объекта), согласованного с преподавателем, создать промпты для 2-3 ИИ-ассистентов (Gigachat, DeepSeek, Qwen, или др.), получить от них код метода, **реализовать метод самостоятельно**, и включить в отчет сравнительный анализ кода, написанного ИИ-ассистентами, и своего варианта. Оценить качество кода, полученного от ИИ-ассистентов, указать на возможные недоработки.
7. Продемонстрировать выполнение программы на сервере helios.
8. Ответить на контрольные вопросы и выполнить дополнительное задание.

Текст, выводящийся в результате выполнения программы не обязан дословно повторять текст, полученный в исходном задании. Также не обязательно реализовывать грамматическое согласование форм и падежей слов выводимого текста.

Стоит отметить, что цель разработки объектной модели состоит не в выводе текста, а в эмуляции объектов предметной области, а именно их состояния (поля) и поведения (методы). Методы в разработанных классах должны изменять состояние объектов, а выводимый текст должен являться побочным эффектом, отражающим эти изменения.

**Требования к объектной модели, сценарию и программе:**

1. В модели должны быть представлены основные персонажи и предметы, описанные в исходном тексте. Они должны иметь необходимые атрибуты и характеристики (состояние) и уметь выполнять свойственные им действия (поведение), а также должны образовывать корректную иерархию наследования классов.
2. Объектная модель должна реализовывать основные принципе ООП - инкапсуляцию, наследование и полиморфизм. Модель должна соответствовать принципам SOLID, быть расширяемой без глобального изменения структуры модели.
3. Сценарий должен быть вариативным, то есть при изменении начальных характеристик персонажей, предметов или окружающей среды, их действия могут изменяться и отклоняться от базового сценария, приведенного в исходном тексте. Кроме того, сценарий должен поддерживать элементы случайности (при генерации персонажей, при задании исходного состояния, при выполнении методов).
4. Объектная модель должна содержать как минимум один корректно использованный элемент каждого типа из списка:
    a. абстрактный класс как минимум с одним абстрактным методом;
    b. интерфейс;
    c. перечисление (enum);
    d. запись (record);
    e. массив или ArrayList для хранения однотипных объектов;
    f. проверяемое исключение.
5. В созданных классах основных персонажей и предметов должны быть корректно переопределены методы equals(), hashCode() и toString(). Для классов-исключений необходимо переопределить метод getMessage().
6. Созданные в программе классы-исключения должны быть использованы и обработаны. Кроме того, должно быть использовано и

обработано хотя бы одно unchecked исключение (можно свое, можно из стандартной библиотеки).

7. При необходимости можно добавить внутренние, локальные и анонимные классы.

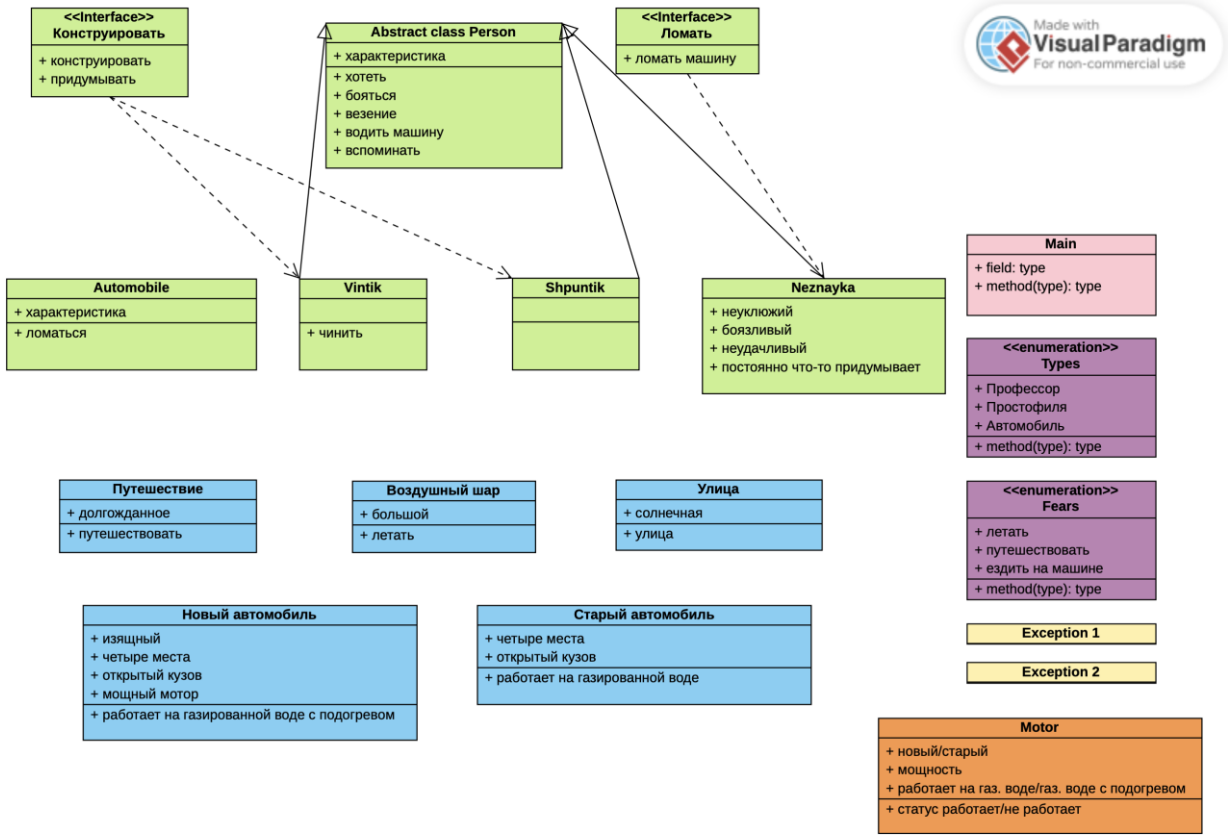
Вариант:

Введите вариант: 32499

**Описание предметной области, по которой должна быть построена объектная модель:**

> Незнайке очень хотелось поскорей отправиться путешествовать, но он вспомнил, как страшно ему было лететь на воздушном шаре, который сконструировал Знайка, и сказал: В это время Незнайка увидел Винтика и Шпунтика, которые проезжали по улице на своем новом автомобиле. Этот автомобиль был четырехместный, с открытым кузовом, такой же, как предыдущий, который Незнайка сломал, когда вздумал покататься на нем. Но, в отличие от предыдущего, этот автомобиль был гораздо изящнее, и мотор у него был более мощный, так как работал он не на обыкновенной газированной воде, а на газированной воде с подогревом.

UML-диаграмма:




Исходный код: https://github.com/ekaterinashukalovich/lab3-programming

Результат работы программы:

1.

```
На солнечной улице появились персонажи:
– Простофиля Незнайка
– Профессор Винтик
– Профессор Шпунтик
– Знайка
Знайка не хотел путешествовать, потому что боится летать

Знайка вспоминает, как он боялся летать
Простофиля Незнайка решил взять автомобиль Винтика и Шпунтика.
Простофиля Незнайка не захотел брать автомобиль.
```

2.

```
На солнечной улице появились персонажи:
– Простофиля Незнайка
– Профессор Винтик
– Профессор Шпунтик
– Знайка
Незнайке хотелось поскорее отправиться в путешествие.

Знайка вспоминает, как он боялся летать
Простофиля Незнайка решил взять автомобиль Винтика и Шпунтика.
Простофиля Незнайка взял автомобиль.
– Мотор: Автомобиль мощностью 60, работает на газированная вода
Попытка поехать на старом автомобиле не удалась: автомобиль сломан и не может ехать
Мотор: Автомобиль мощностью 150, работает на газированная вода с подогревом
Простофиля Незнайка плохо водит и может сломать автомобиль.Профессор Винтик пытается починить машину.Профессор Шпунтик придумал и сконструировал новую машину.
Винтик и Шпунтик едут по улице в новом автомобиле.
```

**Промпт для Qwen:**

Есть Java-класс Незнайка, у которого есть поле driving. Опиши для него метод getDriving, где в случае если Незнайка плохо водит, то машина ломается. Код класса Незнайка не пиши, только код метода.

```
public boolean getDriving() {
   if (!driving) {
      throw new RuntimeException("Машина сломалась: Незнайка плохо водит!");
   }
   return true;
}
```

**Реализованный get.Driving:**

```
try {if (neznayka.getDriving() < 5 || Moves.isBroke(neznayka, vintik) ||
Moves.isBroke(neznayka, shpuntik))
        {
          throw new BadDrivingException(neznayka.getName()
                + " плохо водит и ломает автомобиль." + vintik.getName() +
vintik.construct() + shpuntik.getName() + shpuntik.construct());
        }
```

```
} catch (BadDrivingException e) {
    System.out.println(e.getMessage());
}
```

## Сборка jar-архива на helios

```
mkdir -p classes

javac -d classes -cp . ru/ifmo/se/errors/*.java ru/ifmo/se/stuff/*.java ru/ifmo/se/system/*.java

echo "Main-Class: ru.ifmo.se.system.Main" > manifest.txt

echo "" >> manifest.txt

jar cfm se-app.jar manifest.txt -C classes .

java -jar se-app.jar
```